Sequence Models

Examples of sequence data / tasks:

- sequence recongnition
- Music generation
- Setiment classification
- DNA sequence analysis
- Machine translation
- Video activity recongnition
- Name Entity Recongnition.

} either x, or y, or both are sequence data.
e.g. sequence of words.
sequence of notes...

Notation

Motivating Example:

x: (Harry Potter) and (Hermine Granger) invented a new spell

$x^{<1>}$ $x^{<2>}$ . . . . $x^{<t>}$ $x^{<9>}$ $T_x = 9$

→ y: 1 1 0 1 1 0 0 0 0

$y^{<1>}$ $y^{<2>}$ . . . $y^{<9>}$ $T_y = 9$

$x^{(i)<t>}$: t^{th} element in i^{th} training example

$y^{(i)<t>}$

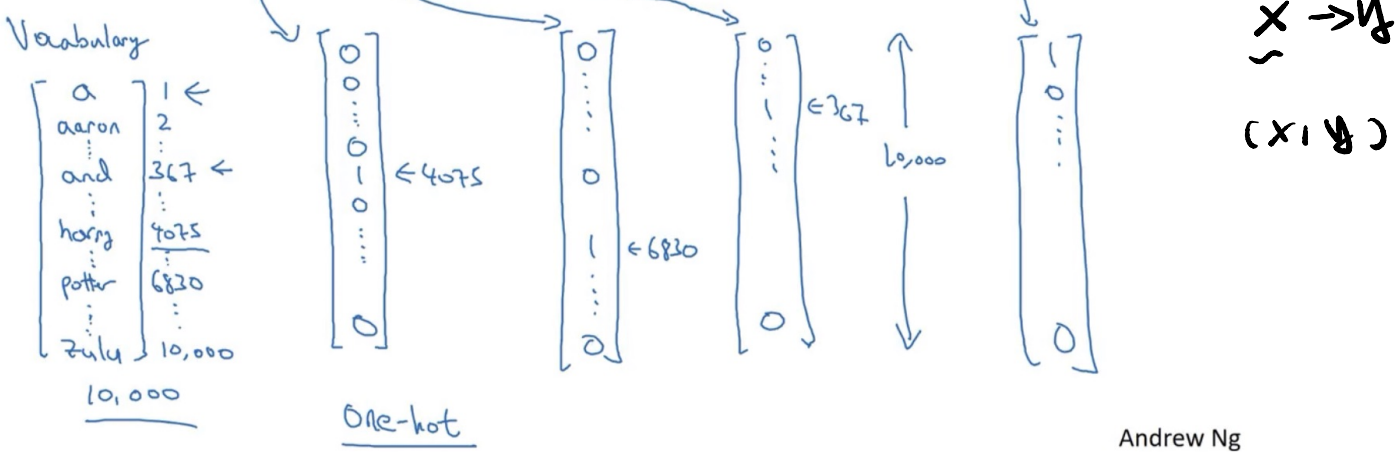
$T_x^{(i)}$: input length for training example i

$T_y^{(i)}$

Representing words

x: Harry Potter and Hermione Granger invented a new spell.

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ ... $x^{<7>}$ $x^{<9>}$

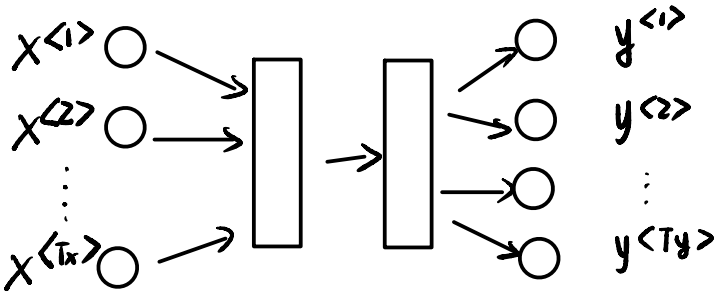


x → y

(x, y)

Recurrent Neural Network model

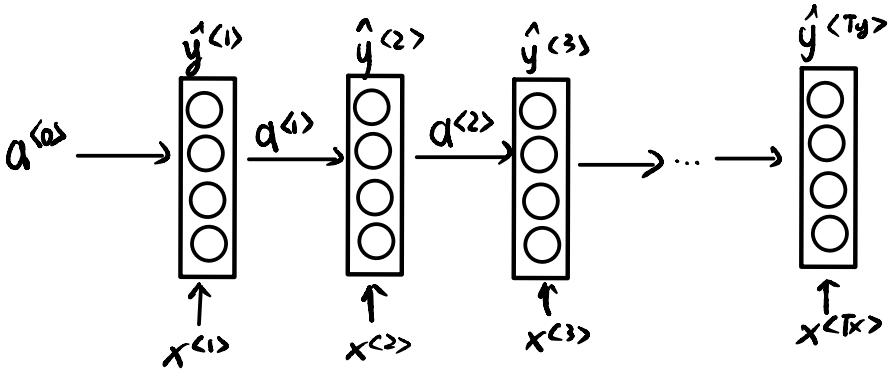
Why standard network doesn't work well with sequence data?



Problems:

- Inputs, outputs can be different lengths in different examples
- Doesn't share features learned across different positions of texts
- Too many weight if input has big shape

RNN



$a^{<0>} = \vec{0}$

$$a^{<1>} = g_1(W_{aa}\,a^{<0>} + W_{ax}\,x^{<1>} + b_a) \leftarrow \tanh / \text{sigmoid}$$

$$y^{<1>} = g_2(W_{ya}\,a^{<1>} + b_y) \leftarrow \text{sigmoid}$$

$$a^{<t>} = g_1(W_{aa}\,a^{<t-1>} + W_{ax}\,x^{<t>} + b_a) \leftarrow \tanh / \text{sigmoid}$$

$$y^{<t>} = g_2(W_{ya}\,a^{<t>} + b_y) \leftarrow \text{sigmoid}$$

Simplified RNN notation

$$a^{<t>} = g_1(W_{aa}\,a^{<t-1>} + W_{ax}\,x^{<t>} + b_a) \longrightarrow a^{<t>} = g(W_a[a^{<t-1>}, x^{<t>}] + b_a)$$

W_{aa}: $(100, 100)$, $a^{<t-1>}$: 100, W_{ax}: $(100, 10{,}000)$, $x^{<t>}$: $10{,}000$

$$W_a = [W_{aa} \mid W_{ax}]: 100 \times 10{,}100$$

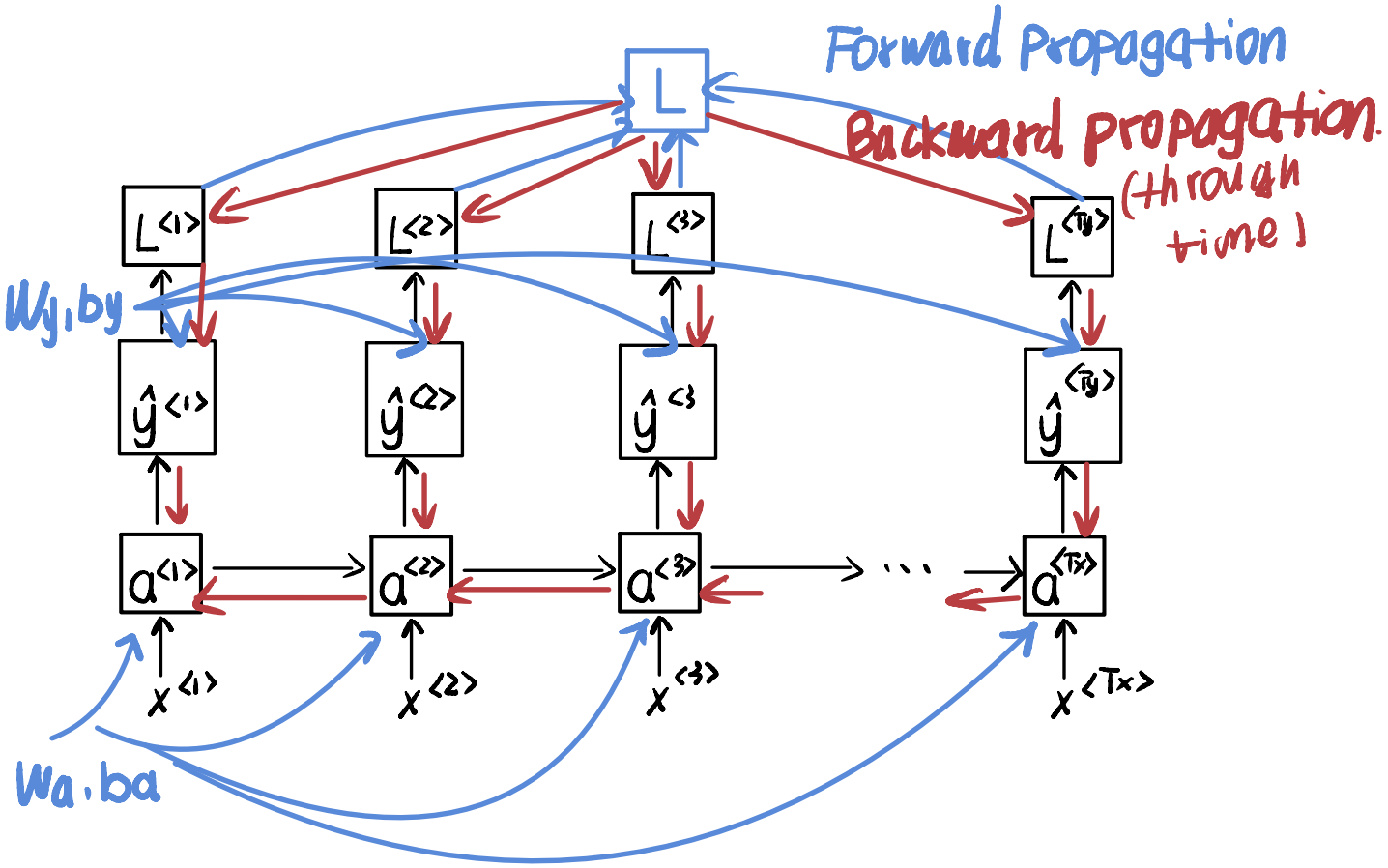
$$\begin{bmatrix} a^{<t-1>} \\ x^{<t>} \end{bmatrix}: 10{,}100$$

$$y^{<t>} = g_2(W_{ya}\,a^{<t>} + b_y) \longrightarrow \hat{y}^{<t>} = g(W_y\,a^{<t>} + b_y)$$

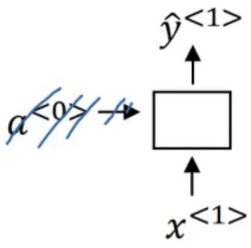
Forward propagation and backpropagation

$$L^{\langle t \rangle}(\hat{y}^{\langle t \rangle}, y^{\langle t \rangle}) = -y^{\langle t \rangle} \log \hat{y}^{\langle t \rangle} - (1 - y^{\langle t \rangle}) \log (1 - \hat{y}^{\langle t \rangle})$$

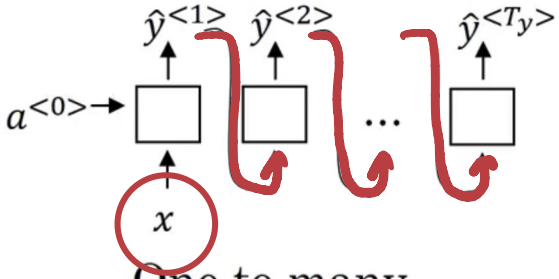
$$L(\hat{y}, y) = \sum_{t=1}^{T_y} L^{\langle t \rangle}(\hat{y}^{\langle t \rangle}, y^{\langle t \rangle})$$



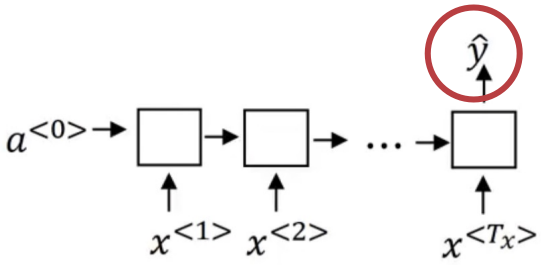
Different Types of RNN



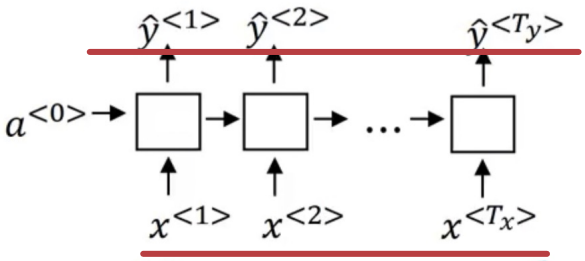
One to one



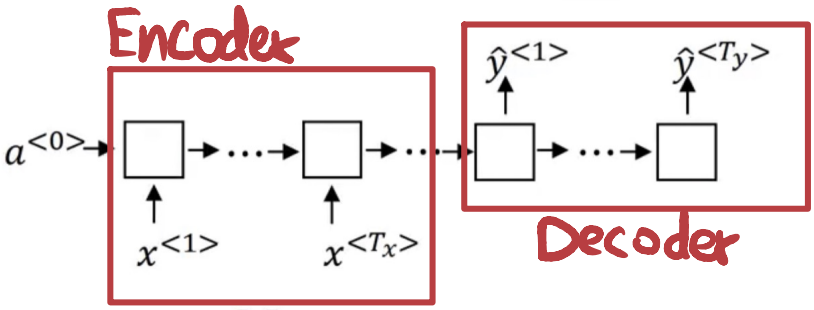
One to many



Many to one



Many to many $T_x = T_y$



Many to many

Andrew Ng

Language model

Example of language modeling:

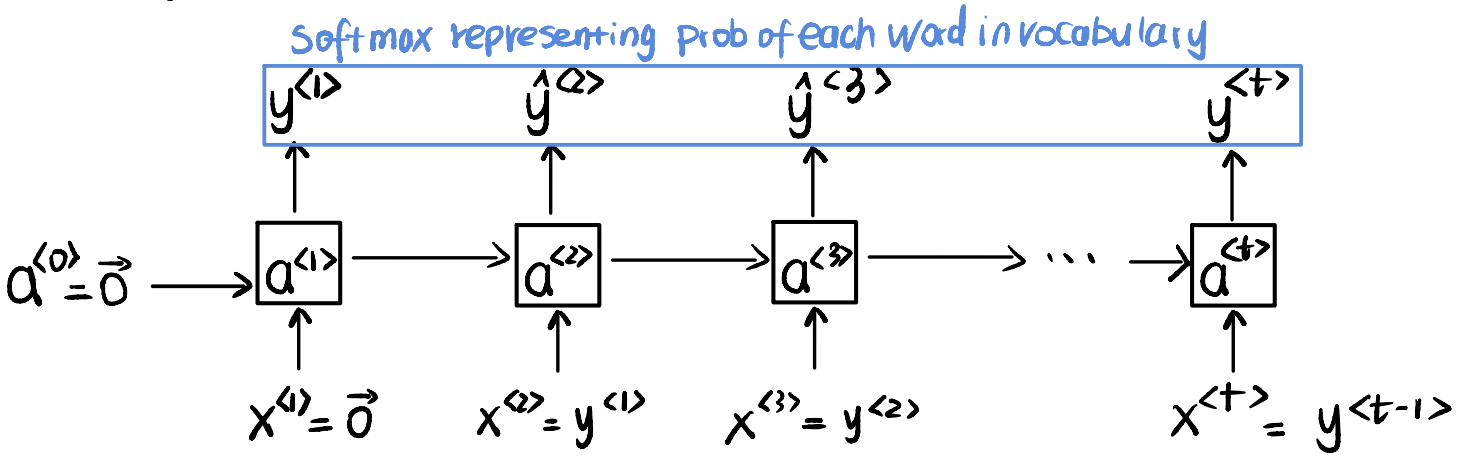
The apple and pair salad $\longrightarrow$ $P(\text{The apple and pair salad}) = 3.2 \times 10^{-13}$

The apple and pear salad $\longrightarrow$ $P(\text{The apple and pear salad}) = 5.7 \times 10^{-10}$

$P(\text{Sentence}) = ?$ $\quad P(y^{<1>}, y^{<2>} \dots y^{<T_y>})$

Language Modeling with RNN

Training set: large corpus of english text



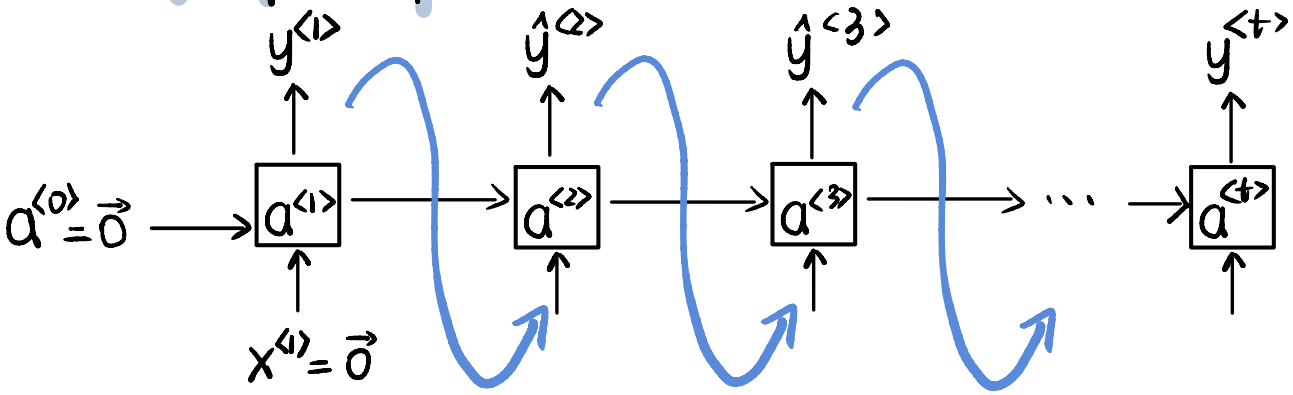
$$\mathcal{L}(\hat{y}^{<t>}, y^{<t>}) = -\sum_i y_i^{<t>} \log \hat{y}_i^{<t>}$$

$$\mathcal{L} = \sum_t \mathcal{L}(\hat{y}^{<t>}, y^{<t>})$$

$$P(y^{<1>}, y^{<2>}, y^{<3>})$$

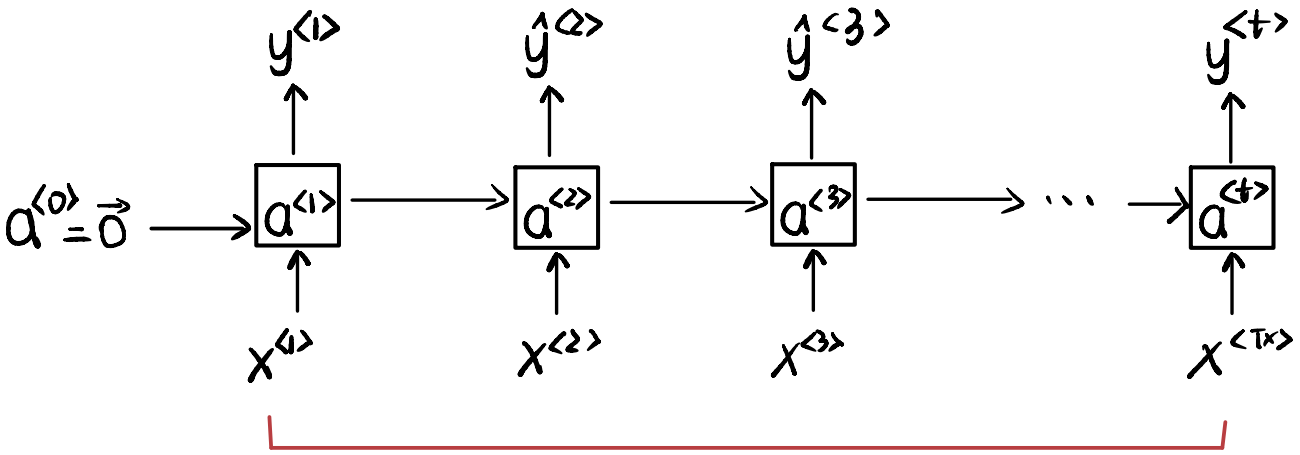
$$= P(y^{<1>})\, P(y^{<2>} \mid y^{<1>})\, P(y^{<3>} \mid y^{<1>}, y^{<2>})$$

Sampling sequence from a trained CNN



GRU And LSTM

Vanishing gradients with RNNs



Depending on input length, this can become very deep.

→ Vanishing / exploding gradients.

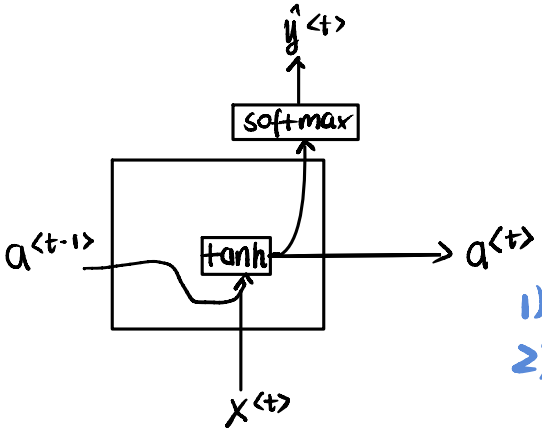
e.g. The cats, which ....., were full
The cat, which ....., was full.

this relation can be buried too deep to be discovered/ remembered by network

In english, this part can be arbitrarily long.

Gated Recurrent Unit (GRU)

Normal RNN unit:



$$a^{<t>} = g(W_a[a^{<t-1>}, x^{<t>}] + b_a)$$

1) Maintain long short term memory
2) Γ_u is normally $\approx 0 \rightarrow c^{<t>} = c^{<t-1>}$, making backpropagation easier (less gradient vanishing)

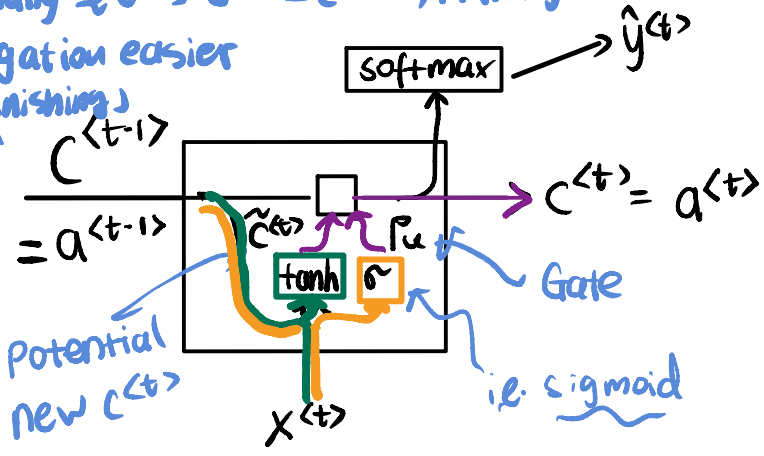
GRU (Simplified):

$$\tilde{c}^{<t>} = \tanh(W_c[c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

↑ element-wise



Full GRU :

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$$

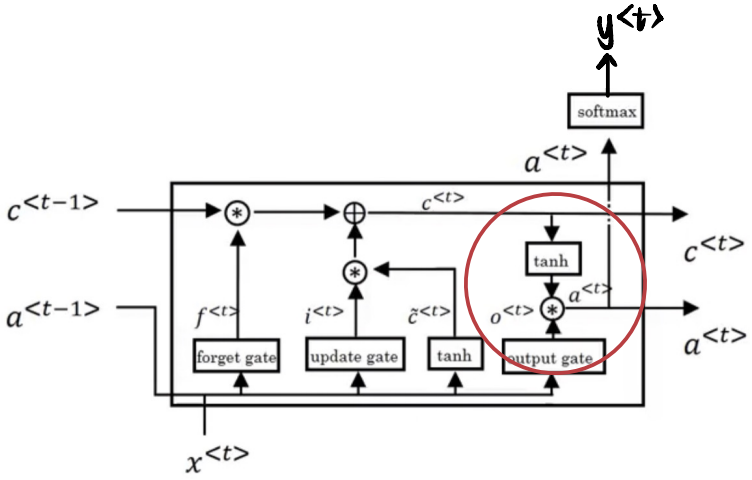
$$\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u)$$

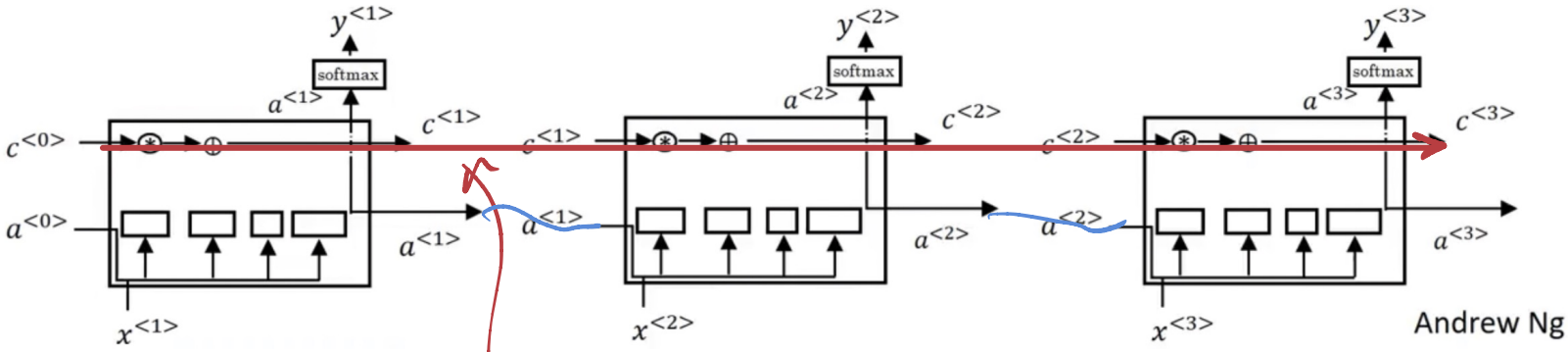
$$\Gamma_r = \sigma(W_r[c^{<t-1>}, x^{<t>}] + b_r)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) + c^{<t-1>}$$

LSTM: long short Term Memory

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$
$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$$
$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$$
$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$$
$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$
$$a^{<t>} = \Gamma_o * \tanh c^{<t>}$$





This acompolishes the long short term Memory

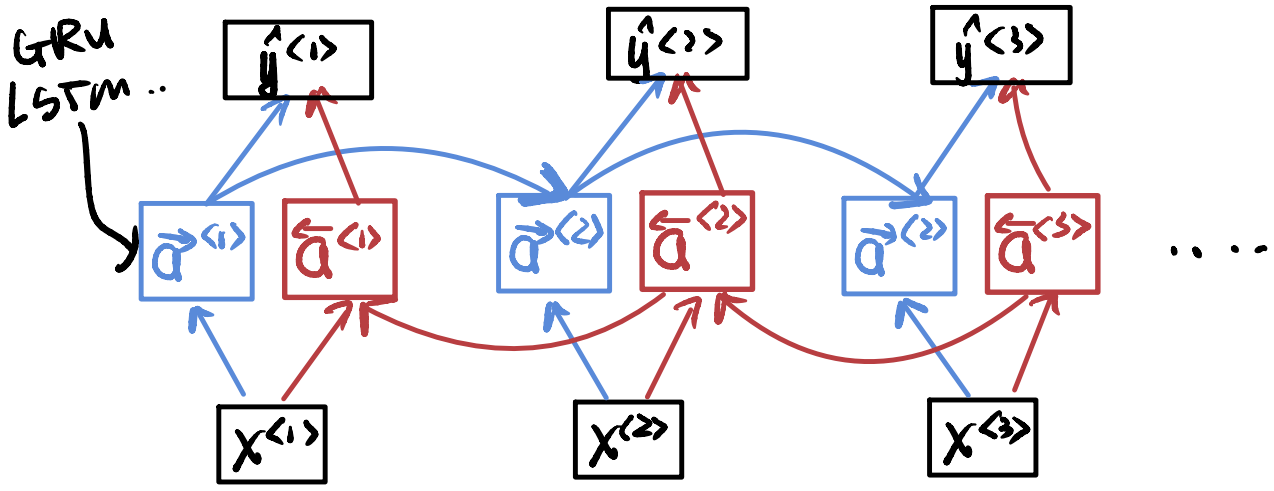
Bidirectional RNN

Motivation:

Unidirectional data is not enough!

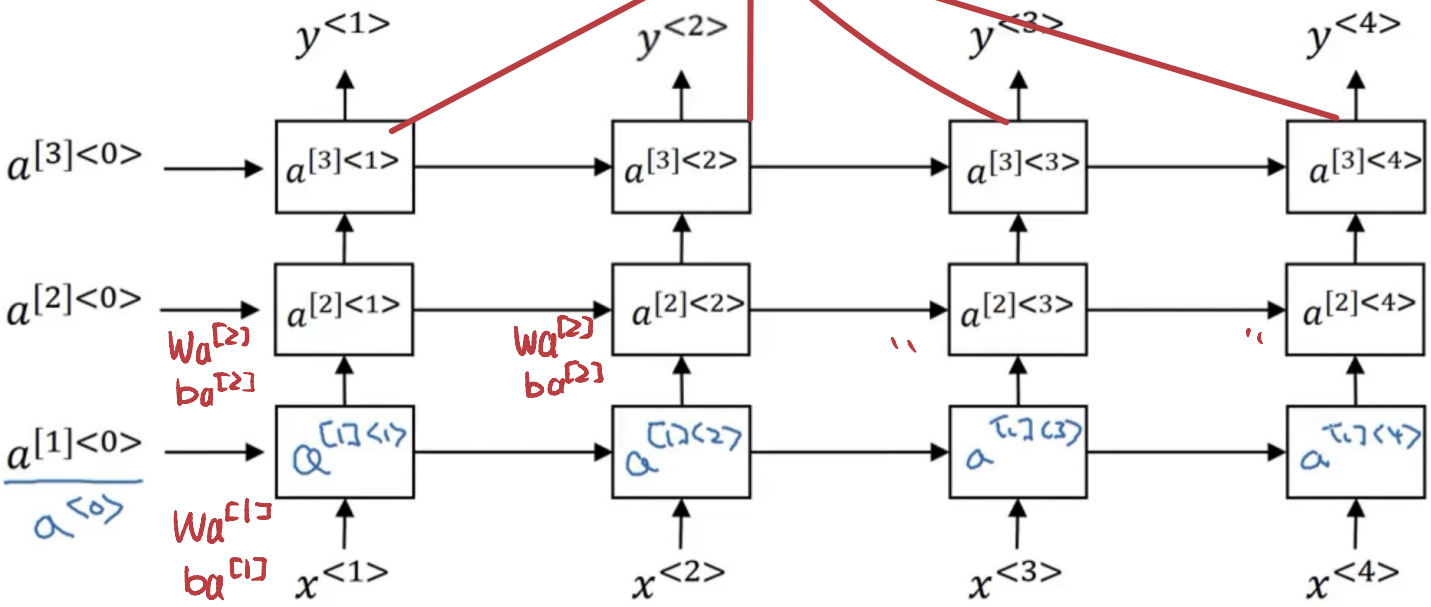
He said, "Teddy bears are on sale!"

He said, "Teddy Roosevelt was a great President!"



Deep RNN

The last few layers can potentially some FC layers



$l=2$, $t=3$

$$a^{[2]<3>} = g(Wa^{[2]}[a^{[2]<2>}, a^{[1]<3>}] + ba^{[2]})$$

Andrew Ng